DAVICOM Semiconductor, Inc.

1588 PTP4L & PTPd 測試, driver 介紹

Stone Shyr

二〇二四年十月十五日

1. PTP4L

1.1. ptp4l is an implementation of the Precision Time Protocol (PTP) according to IEEE standard 1588 for Linux. It implements Boundary Clock (BC) and Ordinary Clock (OC).

1.2. sudo apt install ethtool linuxptp

1.3. Linux kernel base application

(Raspberry Pi 3)

1.4. ptp4l [-AEP246HSLmqsv] [-f config] [-p phc-device] [-l print-level] [-i interface] ...

-A Select the delay mechanism automatically. Start with E2E and switch to P2P when a peer delay request is received.

-E Select the delay request-response (E2E) mechanism. This is the default mechanism. All clocks on single PTP communication path must use the same mechanism. A warning will be printed when a peer delay request is received on port using the E2E mechanism.

-P Select the peer delay (P2P) mechanism. A warning will be printed when a delay request is received on port using the P2P mechanism.

-2 Select the IEEE 802.3 network transport.

-4 Select the UDP IPv4 network transport. This is the default transport.

-6 Select the UDP IPv6 network transport.

-H Select the hardware time stamping. All ports specified by the -i option and in the configuration file must be attached to the same PTP hardware clock (PHC). This is the default time stamping.

-S Select the software time stamping.

-L Select the legacy hardware time stamping.

-f config

Read configuration from the specified file. No configuration file is read by default.

-i interface

Specify a PTP port, it may be used multiple times. At least one port must be specified by this option or in the configuration file.

-p phc-device

(This option is deprecated.) Before Linux kernel v3.5 there was no way to discover the PHC device associated with a network interface. This option specifies the PHC device (e.g. /dev/ptp0) to be used when running on legacy kernels.

-s Enable the slaveOnly mode.

-l print-level

Set the maximum syslog level of messages which should be printed or sent to the system logger. The default is 6 (LOG_INFO).

-m Print messages to the standard output.

-q Don't send messages to the system logger.

-v Prints the software version and exits.

-h Display a help message.

1.5. ~]# ethtool -T eth3

```
     Time stamping parameters for eth3:
     Capabilities:
  hardware-transmit (SOF_TIMESTAMPING_TX_HARDWARE)
  software-transmit (SOF_TIMESTAMPING_TX_SOFTWARE)
  hardware-receive (SOF_TIMESTAMPING_RX_HARDWARE)
  software-receive (SOF_TIMESTAMPING_RX_SOFTWARE)
  software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
  hardware-raw-clock (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
  off (HWTSTAMP_TX_OFF)
  on (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
  none (HWTSTAMP_FILTER_NONE)
  all (HWTSTAMP_FILTER_ALL)
```

1.6. Example:

Master:

```
./ptp4l -m -i eth1
```

```
ptp4l[20351.506]: In clock_create NOW

ptp4l[20351.507]: ++++ config_harmonize_onestep two_step_flag = 0

ptp4l[20351.508]: selected /dev/ptp0 as PTP clock

pi_sync_interval s->kp = 0.300 s->ki 0.100000

ptp4l[20351.512]: port 1 (eth1): INITIALIZING to LISTENING on INIT_COMPLETE

ptp4l[20351.512]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on INIT_COMPLETE

ptp4l[20351.513]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on INIT_COMPLETE

ptp4l[20358.507]: port 1 (eth1): LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES

ptp4l[20358.507]: selected local clock f6444a.fffe.d4b0a7 as best master

ptp4l[20358.507]: port 1 (eth1): assuming the grand master role
```

Slave:

```
./ptp4l -m –s -i eth1
```

2024-08-16-ptp4l-slave-1-step-ok-log.txt

2. PTPd

2.1. Real-Time O.S., LwIP, & ptpd (AT32F437 雅特利)

2.2. PTP daemon (PTPd) is an implementation the Precision Time Protocol (PTP) version 2 as defined by 'IEEE Std 1588-2008'.

PTP provides precise time coordination of Ethernet LAN connected computers.

It was designed primarily for instrumentation and control systems.

2.3. constants.h

DEFAULT_TWO_STEP_FLAG	TRUE
SLAVE_ONLY	TRUE
DEFAULT_DELAY_MECHANISM	E2E
VERSION_PTP	2

2.4. ethernetif.c

void emac_ptptime_gettime(struct ptptime_t * timestamp)

void emac_ptptime_gettime(struct ptptime_t * timestamp)

void emac_ptptime_adjfreq(int32_t adj)

void emac_ptptime_updateoffset(struct ptptime_t * timeoffset)

D:\work\DM9051A\ptpd-2-2-2-at32f437-2-step-2024-03-14-3.txt

3. PTP Linux driver

3.1. PTP hardware clock infrastructure for Linux

A new class driver exports a kernel interface for specific clock drivers and a user space interface. The infrastructure supports a complete set of PTP hardware clock functionality.

+ Basic clock operations

- Set time
- Get time
- Shift the clock by a given offset atomically
- Adjust clock frequency

+ Ancillary clock features

- Time stamp external events
- Period output signals configurable from user space
- Low Pass Filter (LPF) access from user space
- Synchronization of the Linux system time via the PPS subsystem

3.2. Writing clock drivers

Clock drivers include include/linux/ptp_clock_kernel.h and register themselves by presenting a 'struct ptp_clock_info' to the registration method. Clock drivers must implement all of the functions in the interface. If a clock does not offer a particular ancillary feature, then the driver should just return -EOPNOTSUPP from those functions.

Drivers must ensure that all of the methods in interface are reentrant. Since most hardware implementations treat the time value as a 64 bit integer accessed as two 32 bit registers, drivers should use spin_lock_irqsave/spin_unlock_irqrestore to protect against concurrent access. This locking cannot be accomplished in class driver, since the lock may also be needed by the clock driver's interrupt service routine.

3.3. struct ptp_clock_info

Linux kernel 6.11

```
struct ptp_clock_info {
struct module *owner;
char name[PTP_CLOCK_NAME_LEN];
s32 max_adj;
int n_alarm;
int n_ext_ts;
int n_per_out;
int n_pins;
int pps;
struct ptp_pin_desc *pin_config;
int (*adjfine)(struct ptp_clock_info *ptp, long scaled_ppm);
int (*adjphase)(struct ptp_clock_info *ptp, s32 phase);
s32 (*getmaxphase)(struct ptp_clock_info *ptp);
int (*adjtime)(struct ptp_clock_info *ptp, s64 delta);
int (*gettime64)(struct ptp_clock_info *ptp, struct timespec64 *ts);
int (*gettimex64)(struct ptp_clock_info *ptp, struct timespec64 *ts,
                  struct ptp_system_timestamp *sts);
int (*getcrosststamp)(struct ptp_clock_info *ptp,
                      struct system_device_crosststamp *cts);
int (*settime64)(struct ptp_clock_info *p, const struct timespec64 *ts);
int (*getcycles64)(struct ptp_clock_info *ptp, struct timespec64 *ts);
int (*getcyclesx64)(struct ptp_clock_info *ptp, struct timespec64 *ts,
                    struct ptp_system_timestamp *sts);
int (*getcrosscycles)(struct ptp_clock_info *ptp,
                      struct system_device_crosststamp *cts);
int (*enable)(struct ptp_clock_info *ptp,
              struct ptp_clock_request *request, int on);
int (*verify)(struct ptp_clock_info *ptp, unsigned int pin,
              enum ptp_pin_function func, unsigned int chan);
long (*do_aux_work)(struct ptp_clock_info *ptp);
};
```

Linux kernel 5.9.16

```
struct ptp_clock_info {
struct module *owner;
char name[16];
s32 max_adj;
int n_alarm;
int n_ext_ts;
int n_per_out;
int n_pins;
int pps;
struct ptp_pin_desc *pin_config;
int (*adjfine)(struct ptp_clock_info *ptp, long scaled_ppm);
int (*adjfreq)(struct ptp_clock_info *ptp, s32 delta);
int (*adjphase)(struct ptp_clock_info *ptp, s32 phase);
int (*adjtime)(struct ptp_clock_info *ptp, s64 delta);
int (*gettime64)(struct ptp_clock_info *ptp, struct timespec64 *ts);
int (*gettimex64)(struct ptp_clock_info *ptp, struct timespec64 *ts,
                  struct ptp_system_timestamp *sts);
int (*getcrosststamp)(struct ptp_clock_info *ptp,
                      struct system_device_crosststamp *cts);
int (*settime64)(struct ptp_clock_info *p, const struct timespec64 *ts);
int (*enable)(struct ptp_clock_info *ptp,
              struct ptp_clock_request *request, int on);
int (*verify)(struct ptp_clock_info *ptp, unsigned int pin,
              enum ptp_pin_function func, unsigned int chan);
long (*do_aux_work)(struct ptp_clock_info *ptp);
};

****************************************************
strncpy(db->ptp_caps.name,dm9051_ptp__driver_name,sizeof(db->ptp_caps.name));
db->ptp_caps.owner = THIS_MODULE;
db->ptp_caps.max_adj = 50000000;
db->ptp_caps.n_ext_ts = N_EXT_TS;
   db->ptp_caps.n_per_out = N_PER_OUT;
db->ptp_caps.pps = 1;  //Stone add for 1588 pps

db->ptp_caps.adjfine = ptp_9051_adjfine;
//db->ptp_caps.adjfreq = ptp_9051_adjfreq;
db->ptp_caps.adjtime = ptp_9051_adjtime;
```

```
db->ptp_caps.gettime64 = ptp_9051_gettime;
db->ptp_caps.settime64 = ptp_9051_settime;
db->ptp_caps.enable = ptp_9051_feature_enable;
db->ptp_caps.verify = ptp_9051_verify_pin;
****************************************************
```

3.4. Supported hardware

* National DP83640

- 6 GPIOs programmable as inputs or outputs
- 6 GPIOs with dedicated functions (LED/JTAG/clock) can also be used as general inputs or outputs
- GPIO inputs can time stamp external triggers
- GPIO outputs can produce periodic signals
- 1 interrupt pin

Appendix

Appendix 1

Ptp4l => https://manpages.ubuntu.com/manpages/bionic/man8/ptp4l.8.html

Linux driver => https://elixir.bootlin.com/linux/v6.11/source/Documentation/driver-api/ptp.rst

Raspberry Pi about ptp4l => https://github.com/twteamware/raspberrypi-ptp

ptpd+AT32F437 應用 => https://bbs.21ic.com/icview-3254558-1-1.html

ptpd source code => https://github.com/ptpd/ptpd